# PATENT ABSTRACTS OF JAPAN

(11)Publication number : 08-129492
(43)Date of publication of application : 21.05.1996

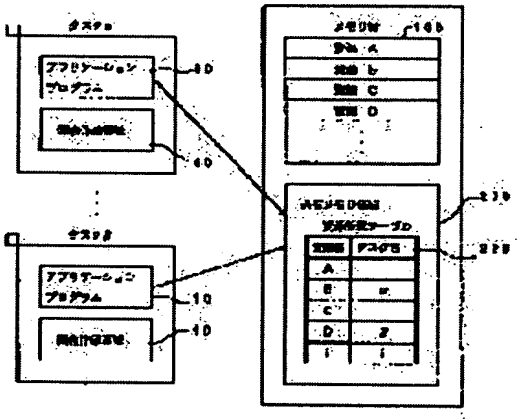
(51)Int.CI. G06F 9/46
G06F 15/00

(21)Application number : 06-268970
(22)Date of filing : 01.11.1994

(71)Applicant : NIPPON TELEGR & TELEPH CORP <NTT>
N T T SOFTWARE KK
(72)Inventor : OKUZAWA OSAMU
WATANABE YOICHI
KUROSAKA SATOSHI

## (54) RESOURCE EXCLUSION CHECKING SYSTEM AND RESOURCE EXCLUSION CHECKING METHOD

(57)Abstract:
PURPOSE: To shorten the confirmation time of an exclusion state by deciding whether resource is secured in a self-task referring to a table preliminarily held by making a pair of a resource name and a task name in advance to the update of resource contents.
CONSTITUTION: This system is composed of tasks α and β having application programs 10 and specific working areas 40, a memory area 100 and a shared memory 200. The shared memory 200 has a resource name and a task name and has a resource control table 220 controlling the using state of resources A, B, C and D respectively. Each resources A, B, C and D arranged on the memory area 100 is sorted by defining the resource name as a key before the tasks α and β are started. When the tasks use resources, a self task name is made a pair with pertinent resources and they are recorded in the resource control table 220, and they are deleted when the uses of them are terminated. Thus, the propriety of the update of the resources by the application program 10 of the tasks in an on-line processing system becomes possible to be decided in short time.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（A）　(11)特許出願公開番号

特開平8－129492

(43)公開日　平成８年(1996)５月21日

| (51)Int.Cl.[6] | 識別記号 | 庁内整理番号 | ＦＩ | 技術表示箇所 |
|---|---|---|---|---|
| Ｇ０６Ｆ　9/46 | ３４０ | Ｆ　7737－5B | | |
| 15/00 | ３１０ | Ｈ　9364－5L | | |

審査請求　未請求　請求項の数５　ＯＬ　（全 11 頁）

(21)出願番号　特願平6－268970

(22)出願日　平成６年(1994)11月１日

(71)出願人　000004226
日本電信電話株式会社
東京都新宿区西新宿三丁目19番２号
(71)出願人　000102717
エヌ・ティ・ティ・ソフトウェア株式会社
神奈川県横浜市中区山下町223番１
(72)発明者　奥沢　修
東京都千代田区内幸町１丁目１番６号　日本電信電話株式会社内
(72)発明者　渡辺　洋一
東京都千代田区内幸町１丁目１番６号　日本電信電話株式会社内
(74)代理人　弁理士　伊東　忠彦

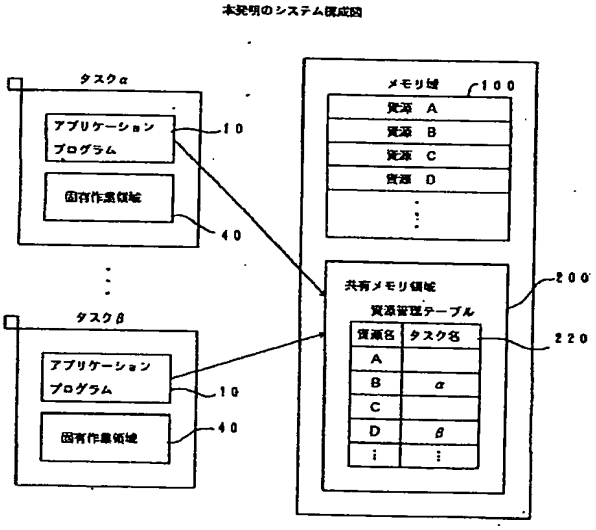
最終頁に続く

(54)【発明の名称】　資源排他チェックシステム及び資源排他チェック方法

(57)【要約】

【目的】　本発明の目的は、排他を必要とする資源にアクセスする際に、排他状態の確認時間を短縮することが可能な資源排他チェック装置及び資源排他チェック方法を提供することである。

【構成】　本発明の資源排他チェックシステムは、タスクの実行に先立って、各タスクが共通にアクセスするメモリ領域内に、管理対象の資源名と資源を使用しているタスク名を組にして保持するテーブルと、各タスクが資源内容を更新する際には、更新に先立って上記テーブルを参照し、資源が自タスクで確保されているか否かを判定する手段を設ける。



1

【 特許請求の範囲】

【 請求項1 】　タスクの実行に先立って、各タスクが共通にアクセスするメモリ領域内に、管理対象の資源名と資源を使用しているタスク名を組にして保持するテーブルと、

各タスクが資源内容を更新する際には、更新に先立って前記テーブルを参照し、該資源が自タスクで確保されているか否かを判定する手段を有することを特徴とする資源排他チェックシステム。

【 請求項2 】　オンライン処理システムの排他を必要とする資源に、タスクがアクセスする際に、該資源の排他状態を確認する資源排他チェック方法において、

該タスクの実行に先立って、各タスクが共通にアクセスするメモリ領域内のテーブルに、アクセスキーを設定し、該アクセスキーで該テーブルをソートしておき、

該タスクによる該アクセスキーに対応する資源の確保状況を該テーブルに記録することを特徴とする資源排他チェック方法。

【 請求項3 】　前記アクセスキーを資源名とし、該資源名と資源を確保しているタスク名の組を前記テーブルに設定し、

アクセス元のタスクから該資源名により資源確保要求が発行されると、前記テーブルを参照し、

アクセスキーに対応する資源が他のタスクにより確保されている場合には、該アクセス元のタスクからのアクセスを排他し、

該アクセスキーに対応する資源がどのタスクからもアクセスされていない場合には、アクセスを受け付け、該テーブルのタスク名に該アクセス元のタスク名を設定する請求項2 記載の資源排他チェック方法。

【 請求項4 】　前記アクセス元のタスクから前記資源名により資源の更新要求が発行されると、前記テーブルを参照し、該アクセスキーに対応する資源が前記アクセス元のタスクにより確保されている場合には、該更新要求を受け付け、確保されていない場合には、エラー情報を出力する請求項3 記載の資源排他チェック方法。

【 請求項5 】　前記タスクの切替、また、前記タスクのアプリケーションプログラムの終了時に、該アプリケーションプログラムで確保していた資源の使用終了を前記テーブルに反映させ、

該資源を他のタスクに解放する請求項3 記載の資源排他チェック方法。

【 発明の詳細な説明】

【０００１】

【 産業上の利用分野】本発明は、排他資源チェックシステム及び資源排他チェック方法に係り、特に、オンライン処理システムにおいて、アプリケーションプログラムがアプリケーションプログラム間で排他制御を必要とする資源にアクセスする際の資源排他チェックシステム及び資源排他チェック方法に関する。

2

【０００２】詳しくは、アプリケーションプログラムが書込み、参照可能なメモリ領域( 資源) にアクセスする場合に、アプリケーションプログラム自体にバグが存在している場合でもメモリの内容を保証する資源排他チェックシステム及び資源排他チェック方法に関する。

【０００３】

【 従来の技術】計算機システム上で動作するソフトウェアは、システムに予め組み込まれているオペレーティングシステム( ＯＳ) と、当該計算機システムの利用目的に合わせて作成されるアプリケーションプログラム( ＡＰ) とに分類できる。ＡＰはＯＳの管理の下で、ＯＳにより割り当てられた資源を利用して動作する。ＯＳはＡＰに割り当てた資源に関してＡＰ間の排他制御は行わない。

【０００４】図7 は、ＡＰが他のＡＰと共用する資源にアクセスする場合のフローチャートである。アプリケーションプログラムＡがメモリ( 資源) を使用する場合に、アクセス対象領域が使用可能かをチェックし( ステップ９１) 、使用可能であれば、当該メモリ領域の使用を宣言する。即ち、該メモリ領域を占有して使用できる状態とする( ステップ９２) 。ＡＰは占有したメモリ領域の内容を更新する( ステップ９３) 。アプリケーションプログラムＡは占有していた資源を解放する。即ち、占有していたメモリ領域を他のＡＰが占有可能な状態に変更する( ステップ９４) 。

【０００５】上記の例を図8 に基づいて説明する。同図( Ａ) は、メモリ領域の占有状況を示し、斜線部分は、アプリケーションプログラムにより占有されている領域を示す。同図( Ｂ) は、メモリ領域を占有及び更新する場合のアプリケーションプログラムＡの例を示す。プログラム例において、ステップ1 からステップ3 において、占有するメモリ領域( ｉ , ｉ ＋ｍ＋１, ｉ ＋３) を指定する。次にステップ4 からステップ6 において、アプリケーションプログラムＡによる更新処理をメモリ領域( ｉ , ｉ ＋３, ｉ ＋ｍ＋１) に対して行い、ステップ7 において、処理が終了すると、使用したメモリ領域を解放する。

【０００６】この様に動作するシステムにおいて、アプリケーションプログラムのバグにより、予め確保していない資源( メモリ領域) にデータを書き込む指示がされる場合がある。例えば、

ＧＥＴ　ｉ：　領域i を確保

ＰＵＴ　ｋ：　領域k ( ｋ ≠ｉ , 領域k は該ＡＰによって予め確保されていない)

この状態でデータ書込み指示を行うような場合である。

【０００７】上記の例で、領域k を他のＡＰが使用している場合( 例えば、他のＡＰが処理結果の一時保持領域としてデータを保持している場合) 、他のアプリケーションプログラムにより書き込まれたデータを破壊してしまうことになる。また、実装されていないメモリ領域が

指定されてしまうことがある。図9 は、タスク内のアプリケーションプログラムの動作を説明するための図である。システムでは、予めタスク毎に割り当てられているアプリケーションプログラムの実行・制御をタスク内の個別作業領域でタスク単位に行う。同図の例では、タスクαとタスクβに同じ通信制御のアプリケーションプログラムXが割り付けられており、タスクαにおいては、アプリケーションプログラムXを用いてタスクαの個別作業領域で月間集計情報の送信処理を行い、タスクβでは、同様にアプリケーションプログラムXを用いてタスクβの個別作業領域で週間集計情報の送信処理を行う。このように、同一のアプリケーションプログラムが複数同時に実行される。このタスクα及びタスクβの個別作業領域は、オペレーティングシステムの制御により、タスクが生成されると、メモリ領域上に確保され、タスクを利用するイベントがなく、タスクが消滅すると、他のタスク向けに解放される。

【０００８】上記のようなタスク間で動作するアプリケーションプログラムがメモリ領域の更新処理を行う場合には、他のタスクによりメモリ領域の更新処理が実行されているか、即ち、対象領域が他のタスクにより確保されているか否かを調べる必要がある。以下にアプリケーションプログラム実行における従来の具体的な資源排他チェックシステムを説明する。

【０００９】図１０は、従来の第1 の資源排他チェックシステムを説明するための図である。同図において、タスクαは、アプリケーションプログラム１０と、アプリケーションプログラム１０の実行により変更されるデータを保持すると共に、資源管理テーブル２１を有し、アプリケーションプログラム１０が使用する全資源を管理する個別作業領域２０及びアプリケーションプログラム１０が使用する固定データを保持するプログラム共通領域３０を有する。資源管理テーブル２１は、アプリケーションプログラム１０が使用する全資源を管理しており、“ＧＥＴ”（領域確保）処理時、ＧＥＴ処理で指定した資源について当該資源管理テーブル２１の管理状況を更新する。同図に示す資源管理テーブル２１では、資源a，ｄが使用されていないことを示し、資源b，ｃが確保されていることを示す。

【００１０】また、タスクαの個別作業領域２０は、タスクの終了とともに、他タスクに解放されるため、タスク起動毎に資源管理テーブル２１の作成処理を行う。資源管理テーブル２１の作成処理は、プログラム共通領域３０より資源管理テーブル２１に複写する、または、対象エントリの個別作業領域２０上のアドレスを取得する処理が必要となる。

【００１１】詳しくは、タスクαの起動後、アプリケーションプログラム１０で使用するメモリ領域（資源）が確保済か否かのチェックを行う場合には、エントリ名に対応する個別作業領域２０の管理テーブル２１のテーブルアドレスをプログラム共通領域３０より読出し、読み出したアドレスで指定される個別作業領域２０から、資源の使用状況情報を取得する。この使用状況情報により、アプリケーションプログラム１０で使用しようとするメモリ領域が自タスクで確保済か否かを判断するものである。



【００１２】図１１は、従来の第2 の資源排他チェックシステムを説明するための図である。同図に示すタスクαは、アプリケーションプログラム１０と資源確保テーブル２２を有する個別作業領域２０を有する。同図に示すシステムは、タスクαで資源を確保する（アプリケーションプログラム１０で“ＧＥＴ”処理を実行する）都度、確保した資源名を、タスクαの個別作業領域２０の資源確保テーブル２２に蓄積・記録しておく。また、アプリケーションプログラム１０により資源内容を更新する（“ＰＵＴ”処理を実行する）場合には、個別作業領域２０の資源確保テーブル２２により更新対象の資源がタスクαによって確保されているか否かを調べ、確保されていれば当該資源の内容を更新する。

【００１３】図１２は、従来の第3 の資源排他チェックシステムを説明するための図である。同図に示す例は、タスクαのアプリケーションプログラム１０によりメモリ領域のアドレスi に資源を確保した（“ＧＥＴ”を実行した）場合に、当該メモリ領域の当該アドレスi にタスク名“α”を書き込んでおき、アプリケーションプログラム１０よりメモリ領域にデータを格納する処理（“ＰＵＴ”）が指定されたことが解釈されると、格納先として指定されたアドレスi の内容を読出し、そこに格納されているタスク名を参照することにより、当該タスクで当該メモリ領域が確保されているかを確認した後に、データ書込みを実行する。なお、この方法を行う場合には、メモリ領域のどの領域にタスク名を書き込んだのかを前述の第1 の例または、第2 の例のテーブル２１、２２に書き込んでおくものとする。

【００１４】

【発明が解決しようとする課題】しかしながら、上記各従来の方法は、以下のような問題点がある。まず、第1 の従来の方法は、タスク上で走行するアプリケーションプログラムが対象とする全てのメモリ領域（資源）を管理するテーブルをタスク毎に用意する必要があり、大容量の個別作業領域が必要となり、タスクを実行する毎に必要となるメモリ量が増大するという問題がある。特に、管理対象とする資源数が多い場合、例えば、資源としてシステム利用者向けにデータを配信するために『各利用者毎のデータ格納アドレス』のエントリの場合には、利用者数分の資源数が必要となり、システム利用者数が数千～数万に及ぶ場合には、膨大な領域が必要となる。

【００１５】さらに、個別作業領域はタスク起動の都度、タスク毎に割り付けられるが、アプリケーションプ

5

ログラム、プログラム共通領域はタスク共通にメモリ領域上に割り当てられ、同一のプログラムを実行するタスクが複数個同時に起動される場合( マルチタスク処理) には不都合が生じる。次に、第2 の従来の方法において、資源確保テーブルには、確保した順序に検索対象の資源名が記録されているため、当該資源が確保されているか否かについては、検索の都度、資源確保テーブルを先頭から順にサーチすることになり、処理時間が長いという問題がある。また、第2 の従来の方法において、資源確保テーブルの内容を資源名を検索キーとしてソート処理を行うことは、以下の2 点により困難である。

【００１６】１つには、アプリケーションプログラムでは、“ＧＥＴ”( 確保処理) 、“ＰＵＴ”( 更新処理) が交互に出現することがあるため、ソート処理の適切な時期を決定できない。もう１つには、ソート対象のエントリ数が数千、数万等と多い場合、ソート処理は一般的に長時間を要し、タスク実行中にソート処理を行うと、タスクの実行時間が長くなり、レスポンス時間が長くなるため、リアルタイムで処理を行うオンライン処理への適用は好ましくない。

【００１７】最後に、第3 の従来の方法は、アプリケーションプログラムに対して用意された資源、例えば、メモリ領域を制御側が使用することになり、アプリケーションプログラムの設計が困難である。また、資源内の各更新処理における、当該資源の更新可能か否かのチェック時、及び、資源確保処理(“ＧＥＴ”処理) 時に、実際にアクセス対象資源にアクセスすることから、チェック時間を要する。また、タスク終了処理時にアクセス対象資源の排他( 解放) ・解除を行う必要性から資源名を順次読み込み、資源解放処理を行うため、スタックあるいは、タスク名等で対象資源領域を全サーチする必要がある。従って、サーチに長時間を要するという問題がある。

【００１８】本発明は、上記の点に鑑みなされたもので、上記従来の問題点を解決し、排他を必要とする資源にアクセスする際に、排他状態の確認時間を短縮することが可能な資源排他チェックシステム及び資源排他チェック方法を提供することを目的とする。本発明の更なる目的は、アプリケーションプログラムのバグ等により当該アプリケーションプログラムで確保していない資源の内容を更新するような動作がプログラム上で指示された場合であっても、更新処理の可否をチェックし、確保していない資源の更新処理を禁止することが可能な資源排他チェックシステム及び資源排他チェック方法を提供することである。

【００１９】

【課題を解決するための手段】図1 は、本発明の原理を説明するための図である。本発明の資源排他チェックシステムは、上記各問題点を解決するため、タスクの実行に先立って、各タスクが共通にアクセスするメモリ領域

6

内に、管理対象の資源名と該資源を使用しているタスク名を組にして保持するテーブルと、各タスクが資源内容を更新する際には、更新に先立って上記テーブルを参照し、該資源が自タスクで確保されているか否かを判定する手段を設ける。

【００２０】また、本発明の資源排他チェック方法は、アクセスキーを資源名とし、資源名と資源を確保しているタスク名の組をテーブルに設定し、アクセス元のタスクからアクセスされた時、テーブルを参照し、アクセスキーに対応する資源が他のタスクにより確保されている場合には、アクセス元のタスクからのアクセスを排他し、アクセスキーに対応する資源がアクセス元のタスクからアクセスされている場合には、アクセスを受け付け、テーブルのタスク名にアクセス元のタスク名を設定する。

【００２１】また、本発明は、アクセス元のタスクから資源名により資源の更新要求が発行されると、テーブルを参照し、アクセスキーに対応する資源がアクセス元のタスクにより確保されている場合には、更新要求を受け付け、確保されていない場合には、エラー情報を出力する。また、本発明は、タスクの切替、または、タスクのアプリケーションプログラムの終了時に、アプリケーションプログラムで確保していた資源の使用終了をテーブルに反映させ、資源を他のタスクに解放する。

【００２２】

【作用】本発明は、タスクの実行前、即ち、システム起動時に、タスクからの資源確保状態を記録するテーブルに、被アクセス側の資源名を予め設定するとともに、プログラムが走行する前にバッチ処理で資源名によりソートしておく。また、タスク( アプリケーションプログラム) が資源を使用する場合に、自タスク名を該当する資源と対にしてテーブルに記録し、使用終了時に消却する。これにより、オンライン処理において、タスクのアプリケーションプログラムによる資源の更新の可否を短時間で判定することが可能である。

【００２３】また、本発明は、ソート処理済にテーブルに対して、タスクからの資源確保状況を問い合わせ、他のタスクで当該資源を確保している場合には、アクセスを無効とし、自タスクで資源を確保している場合のみ自タスクの処理を実行するものである。従って、アプリケーションプログラムのバグ等により、未確保の資源に対するアクセスを防止する。

【００２４】

【実施例】以下、本発明の実施例を詳細に説明する。図2 は、本発明のシステム構成を示す。同図に示すシステムは、アプリケーションプログラム１０、固有作業領域４０を有するタスクα、β、複数の資源を有するメモリ域１００と、タスクが共通してアクセスする共有メモリ領域２００を有するメモリより構成される。共有メモリ領域２００は、各資源Ａ，Ｂ，Ｃ，Ｄの使用状態を管理

7

する資源管理テーブル２２０を有する。

【００２５】資源管理テーブル２２０は、資源名とタスク名の項目からなり、アクセスキーとなる資源名には、メモリ域１００に配置されている各資源名Ａ，Ｂ，Ｃ，Ｄが設定されており、タスクα、βが起動される前にこの資源名をキーとしてソートされている。同図の例ではアルファベット順に昇順にソートされているものとする。また、資源名に対応する項目として、タスク名がある。タスク名は、当該資源名に対応する資源がタスクにより確保されている場合に、確保しているタスク名が設定される。図2の例において、資源管理テーブル２２０のタスク名の欄には、資源名Ｂの資源を確保しているタスクとして、タスクαが設定されており、資源名Ｄの資源を確保しているタスクとしてタスクβが設定されている。

【００２６】図3は、本発明の一実施例のタスクの構成を示す。同図に示すタスクは、アプリケーションプログラム１０とタスクに固有の作業領域４０より構成され、固有作業領域４０は、アクセス履歴テーブル４１を有する。アクセス履歴テーブル４１は、アプリケーションプログラム１０の命令により、資源管理テーブル２２０にアクセスした順に資源名を記録していく。また、資源管理テーブル２２０を参照し、資源を確保できた場合には、アクセス履歴テーブルの資源名のフラグをオンにする。図3の例では、資源Ｃが当該タスクにより確保されているものとして“＊”を付与している。また、アプリケーションプログラム１０が終了して、資源を他のタスクに解放した場合には、アクセス管理テーブル２２０の資源名に対応するタスク名を消去すると共に、アクセス履歴テーブル４１のフラグはオフとし、図3の例においては、当該資源名に対応するフラグ欄を空白とする。

【００２７】次に、上記のシステム構成においてタスク側から資源をアクセスする動作を説明する。図4は、本発明の一実施例の動作の概要を説明するためのフローチャートである。まず、バッチ処理として、資源管理テーブル２２０にタスクで使用する全ての資源の資源名を設定し（ステップ１０１）、資源名をキーにして資源管理テーブル２２０の項目をソートしておく（ステップ１０２）。

【００２８】オンライン処理のタスクが起動される（ステップ１０３）。なお、ここで、アプリケーションが終了した場合には（ステップ１０４）、資源管理テーブル２２０において以下に述べる資源確保、資源内容更新処理等が行われた資源名に対応するタスク名を消去し（ステップ１０５）、処理を終了する。アプリケーションプログラムを実行し（ステップ１０６）、その命令が資源確保要求命令か、資源内容更新命令かを判定し（ステップ１０７）、資源確保要求命令であれば、自タスクの個別作業領域４０にアクセスする資源名を記録する（ステップ１０８）。次に、資源管理テーブル２２０を資源名

8

をキーとしてサーチし（ステップ１０９）、このとき、当該資源名に対応する利用中タスク名に他のタスク名が記載されていれば（ステップ１１０、ｙｅｓ）、当該資源は、他のタスクで使用されている状態を意味するので、ステップ１０４に処理に移行し、他のタスクが使用していなければ（ステップ１１０，ｎｏ）、資源管理テーブル２２０の資源名にタスク名を対応付けて記録する（ステップ１１１）。

【００２９】また、ステップ１０７において、アプリケーションプログラムの命令が資源内容更新命令であれば、資源管理テーブル２２０を資源名をキーにして参照して、自タスクで確保済であるかを確認し（ステップ１１２）、自タスクで確保している資源であれば、資源名に対応する資源を更新する（ステップ１１３）。また、自タスクで確保済となっていない資源であれば、エラーを出力する（ステップ１１４）。

【００３０】図5は、本発明の一実施例の動作を説明するための図である。

（１）まず、資源管理テーブル２２０が共有メモリ領域２００内に設定され、資源名でＡ→Ｂ→Ｃ→Ｄの順にソートする。

（２）タスクαが起動され、アプリケーションプログラムａの処理が開始される。アプリケーションプログラムａの最初の命令は『ＧＥＴ　Ｂ』である。この命令は、資源Ｂを確保するための命令であるので、まず、タスクα内のアクセス履歴テーブル４１に資源名“Ｂ”を書込み、資源管理テーブル２２０の資源名“Ｂ”をサーチして資源確保要求を発行する。このとき、資源管理テーブル２２０の資源“Ｂ”は、どのタスクからも確保されていないので、アプリケーションプログラムａでこの資源を確保することが可能である。従って、アプリケーションプログラムａでは資源名“Ｂ”を確保し、タスク名『α』を資源管理テーブル２２０の資源名“Ｂ”に対応する欄に書き込む。さらに、タスクα内のアクセス履歴テーブル４１の資源名“Ｂ”に対応するフラグをオンにする。

【００３１】（３）次に、タスクαのアプリケーションプログラムａの命令は、『ＰＵＴＢ，ｘ』であるので、再度資源管理テーブル２２０を参照する。ここで、同じタスクαにより資源“Ｂ”が確保されているので、この命令が実行できる。従って、資源“Ｂ”のメモリ領域の内容を「ｘ」に更新することができる。

（４）さらに、タスクαのアプリケーションプログラムａの命令は、『ＧＥＴＡ』であるので、タスクαのアクセス履歴テーブル４１に資源名“Ａ”を書込み、資源名“Ａ”をキーとして、資源管理テーブル２２０をサーチし、資源確保要求を発行する。このとき、資源管理テーブル２２０の資源“Ａ”は、どのタスクからも確保されていないことが分かる。従って、アプリケーションプログラムａはこの資源を使用することが可能であるので、

9

資源“Ａ”を確保し、タスク名『 α』を資源管理テーブル２２０の資源名“Ａ”に対応する欄に書き込む。さらに、タスクαのアクセス履歴テーブル４１のフラグをオンにする。

【００３２】（５）次の命令は、『ＰＵＴ　Ａ，ｙ』であるので、資源名“Ａ”をキーとして資源管理テーブル２２０をサーチする。このとき、資源名“Ａ”が自タスクαにより確保されているので、資源名“Ａ”の内容を「ｙ」に更新することができる。

（６）次に、タスクβが起動され、アプリケーションプログラムｂの処理が開始される。アプリケーションプログラムｂの最初の命令は、『ＧＥＴ　Ｄ』であり、資源名“Ｄ”を確保するための命令である。従って、タスクβのアクセス履歴管理テーブル４１に資源名“Ｄ”を記入し、資源名“Ｄ”をキーとして資源管理テーブル２２０をサーチし、資源確保要求を発行する。このとき、資源管理テーブル２２０の資源“Ｄ”はどのタスクからも確保されていないことが分かる。従って、資源“Ｄ”を確保し、タスク名『 β』を資源管理テーブル２２０の資源名“Ｄ”に対応する欄に書き込む。このとき、タスクβのアクセス履歴テーブル４１の資源名“Ｄ”に対応するフラグをオンにする。

【００３３】（７）さらに、タスクβのアプリケーションプログラムｂの命令は、『ＰＵＴＣ』であるので、資源名“Ｃ”をキーとして、資源管理テーブル２２０をサーチし、更新要求を発行する。このとき、資源管理テーブル２２０の資源“Ｃ”は、タスクβにより確保されていないため、命令『ＰＵＴ　Ｃ』に対してはエラーを返す。

【００３４】（８）ここで、アプリケーションプログラムａ，ｂの処理が終了したら、資源管理テーブル２２０の利用中タスク欄に書き込まれたタスク名を消去し、さらに、各タスクの固有作業域４０のアクセス履歴管理テーブル４１のフラグ欄を全てオフにする。

上記の（７）に示すように、アプリケーションプログラムｂには、領域Ｄに対して『ＰＵＴ　Ｃ』命令を発行する前に『ＧＥＴ　Ｃ』は発行していないにも関わらず、メモリ領域を更新する命令を発行しているため、このアプリケーションプログラムｂには、バグがあることがわかる。このように、資源管理テーブル２２０を参照することにより、ある資源にメモリ領域を確保していないまま、命令が実行されることを防止する。

【００３５】さらに、以下に他の実施例を説明する。図６は、本発明の他の実施例の排他チェックの動作を説明するための図である。同図に示す例は、メモリ領域１００に利用者テーブル６００を設けた例である。システムの利用が予め限定されているという設定である場合には、システムで１つ、利用者Ｉ Ｄを管理するテーブル（以下、利用者テーブル）６００を用意し、利用者Ｉ Ｄに基づいてソートしておくことが可能である。

10

【００３６】また、受け付け処理では、受け付け順に利用者Ｉ Ｄをスタックするのみならず、利用者テーブル６００に対して、当該タスクで使用される旨を記録しておく。利用者テーブル６００へのアクセスは、キーによるバイナリサーチを行って、確認する。出力依頼時には、該当タスクで処理されたかどうかを利用者テーブル６００をバイナリサーチして確認する。これにより、サーチ時間が短縮されることから、出力依頼処理時間の短縮が見込まれる。

【００３７】なお、終了処理時には、登録受け付け順にスタックされた利用者Ｉ Ｄをシーケンシャルに読出し、利用者Ｉ Ｄをキーとして利用者テーブル６００の消去を行うことも可能である。なお、本発明は、上記の実施例に限定されることなく、特許請求の範囲内で種々変更・応用が可能である。

【００３８】

【発明の効果】上述のように、本発明によれば、アプリケーションプログラム用資源管理テーブルを、タスクの実行の都度ではなく、全てのタスクの実行に先立って（例えば、システム生成時）生成しておくことが可能であるので、オフライン処理として、アクセスキーである資源名によるソート処理を実行できる。即ち、オンライン処理には影響を与えずに、資源管理テーブルの内容をソートすることができる。

【００３９】このように、タスクの実行に先立って、ソート処理が可能であるため、排他を必要とする資源にアクセスする際に、毎回同じように排他状態の確認をシーケンシャルサーチにより行う必要がないため、資源の排他制御状況を確認するための時間が短縮される。さらに、本発明によれば、アプリケーションプログラムのバグ等により当該アプリケーションプログラムで確保していない資源の内容を更新するような動作がプログラム上で指示されていた場合でも更新の可否をチェックし、資源を確保していない限り更新処理ができないような制御が可能である。

【００４０】また、本発明は、アプリケーションプログラム用資源管理テーブルがアクセスキーにより、予めソートされているので、従来の技術と比較して、更新処理の可否の判定時間を短縮することでき、特に、対象資源量が多い場合には特に有効である。また、各タスク、各アプリケーションプログラム共通に資源管理テーブルを持つため、多大のメモリ量を必要としない。

【図面の簡単な説明】

【図1】本発明の原理を説明するための図である。

【図2】本発明のシステム構成図である。

【図3】本発明の一実施例のタスクの構成図である。

【図4】本発明の一実施例の動作の概要を説明するためのフローチャートである。

【図5】本発明の一実施例の動作を説明するための図である。

11

【 図6 】本発明の他の実施例の排他チェックの動作を説明するための図である。

【 図7 】ＡＰが他のＡＰと共用する資源にアクセスする場合のフローチャート である。

【 図8 】従来の資源排他処理を説明するための図である。

【 図9 】タスク内のアプリケーションプログラムの動作を説明するための図である。

【 図１０ 】従来の第1 の資源排他チェックシステムを説明するための図である。

【 図１１ 】従来の第2 の資源排他チェックシステムを説

12

明するための図である。

【 図１２ 】従来の資源排他チェックシステムを説明するための図である。

【 符号の説明】

１０　アプリケーションプログラム

４０　固有作業領域

４１　アクセス履歴テーブル

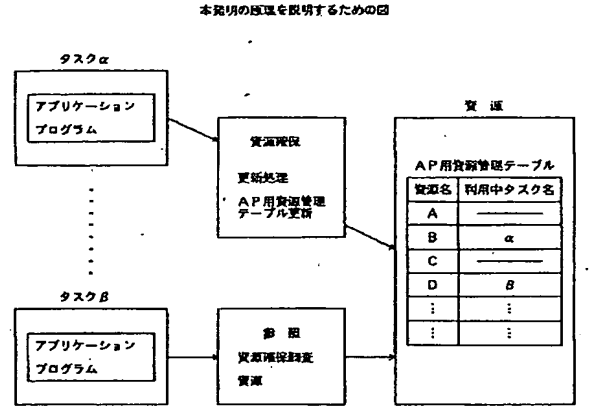
１００　メモリ 領域

２００　共有メモリ 領域

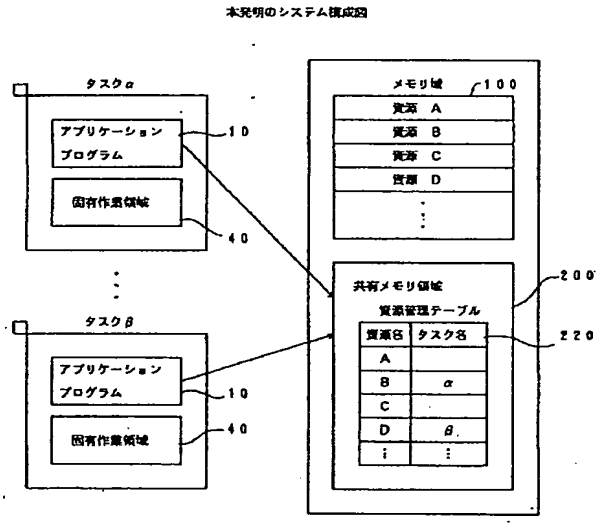
10　２２０　資源管理テーブル

６００　利用者テーブル

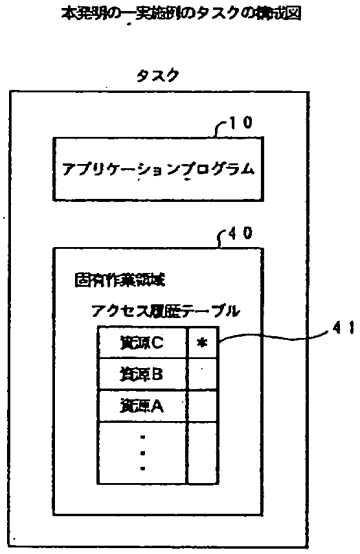
【 図1 】



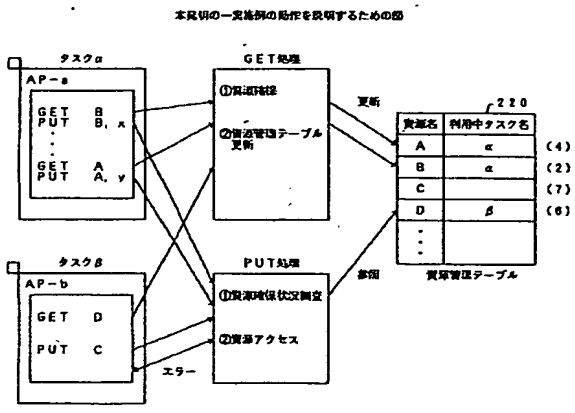

【 図2 】




【 図3 】

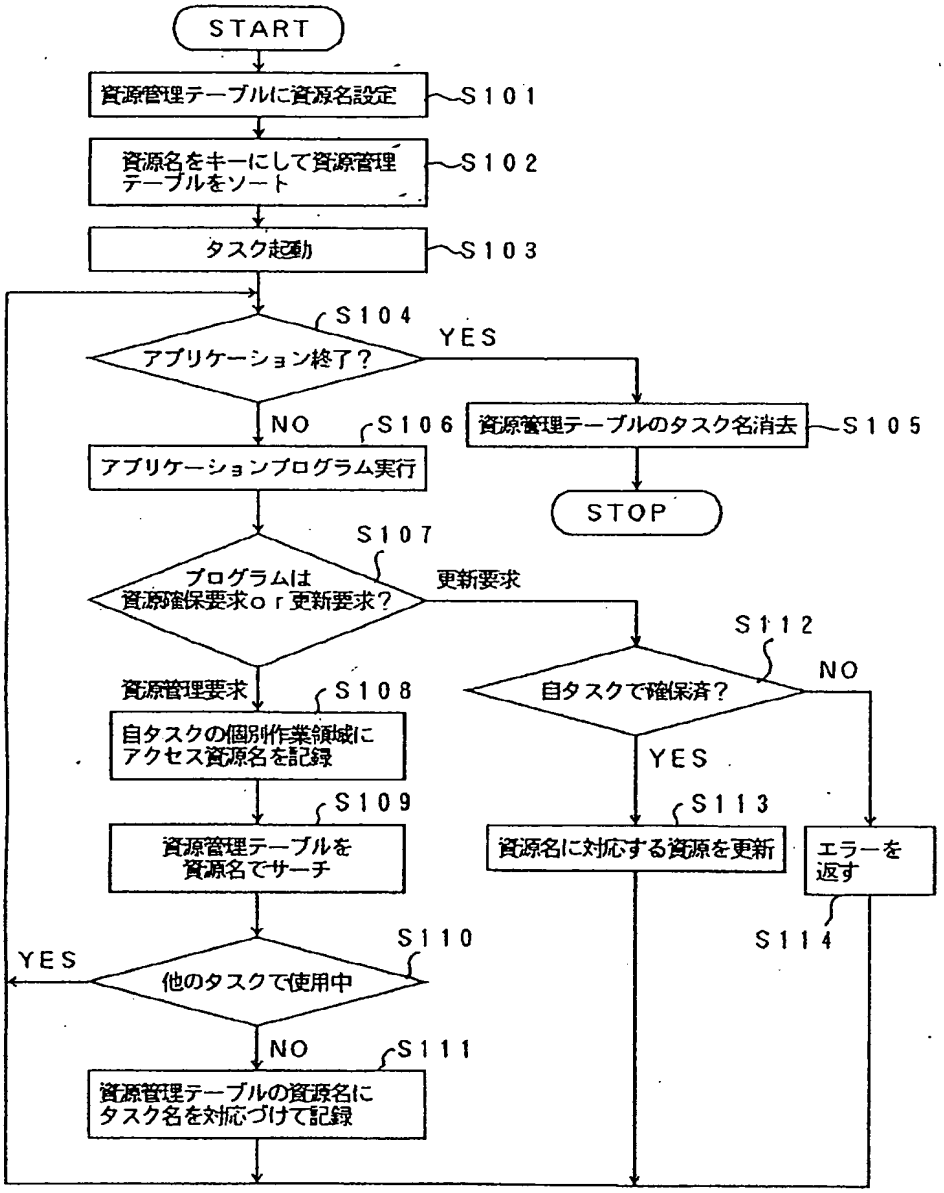



【 図5 】



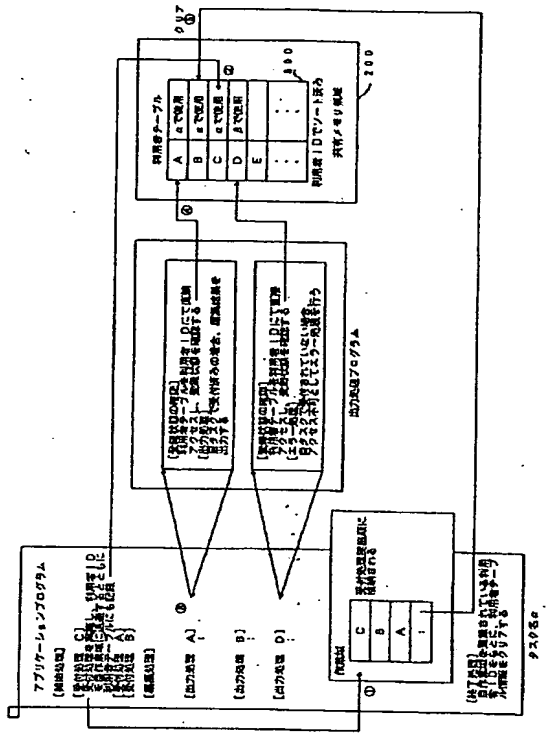
【 図4 】

本発明の一実施例の動作の概要を説明するためのフローチャート



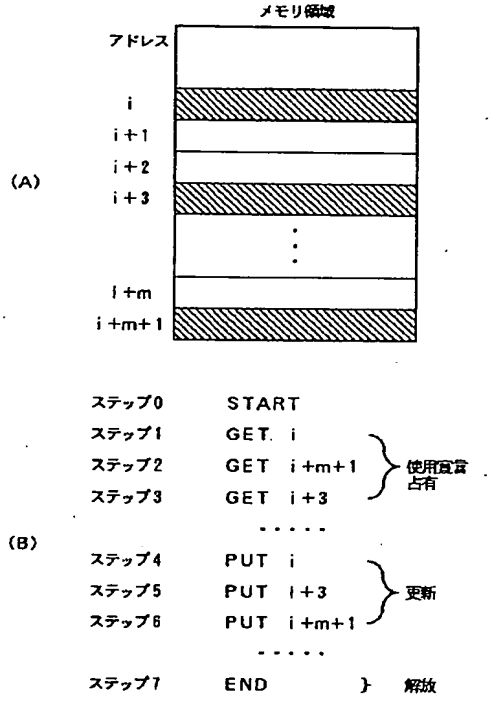
【 図6 】

本発明の他の実施例の排他チェックの動作を説明するための図



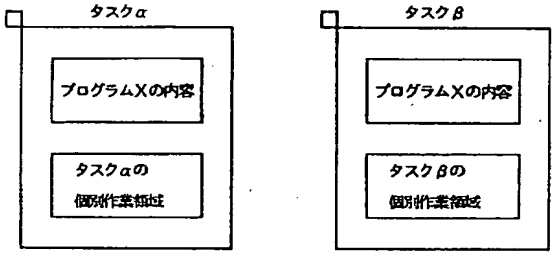
【 図8 】

従来の資源排他処理を説明するための図



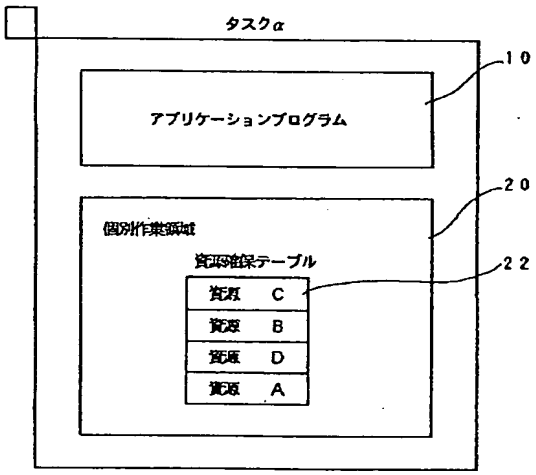

【 図9 】

タスク内のアプリケーションプログラムの動作を説明するための図



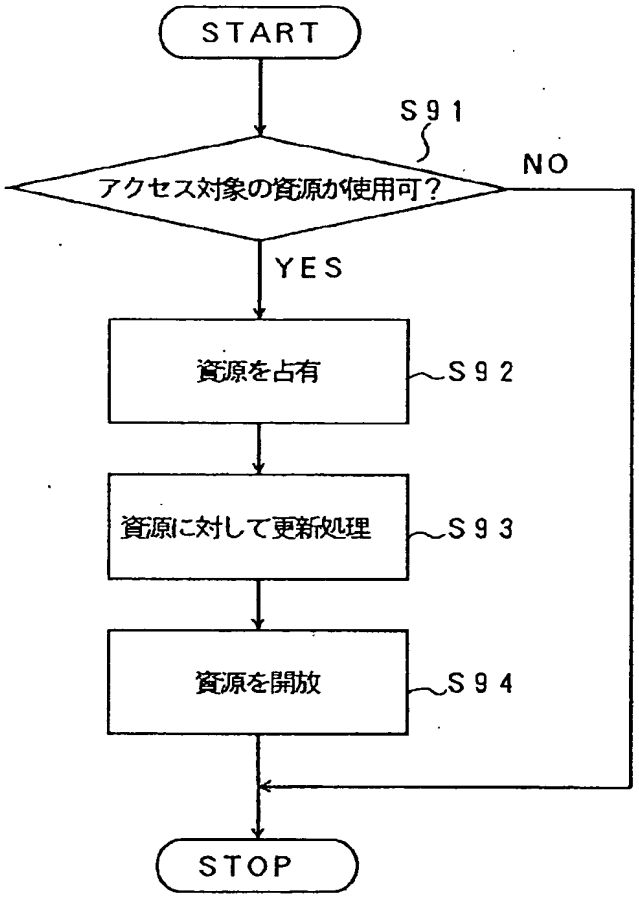

【 図11 】

従来の第２の資源排他チェックシステムを説明するための図

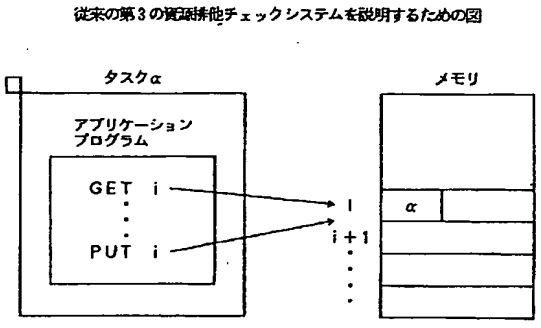


【図7】

APが他のAPと共用する資源にアクセスする場合のフローチャート

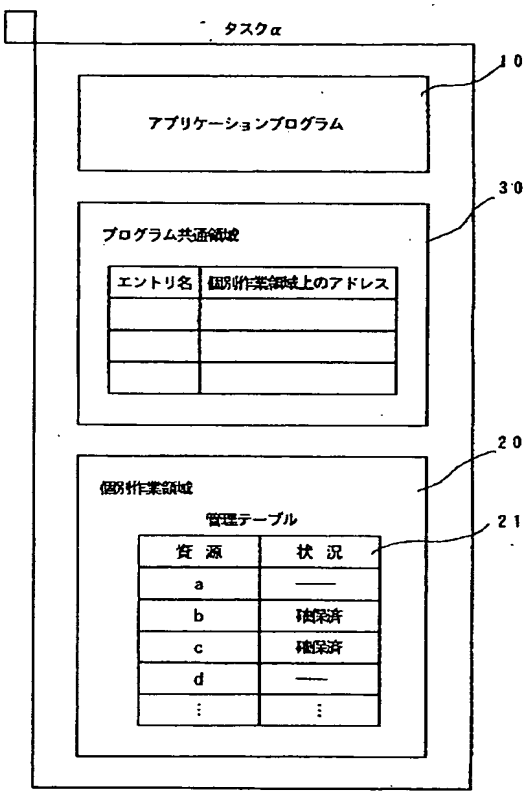



【図12】



【図１０】

従来の第１の資源排他チェックシステムを説明するための図




フロントページの続き

(72)発明者　黒坂　聡
神奈川県横浜市中区山下町223番1 号　エヌ・ティ・ティ・ソフトウェア株式会社内